University of Texas at Arlington

CSE-3330-002

April 17, 2020

**Flight Reservation Project - Phase 2**

**Database Creation and Querying**

**In MySQL**

**Team 02**

Ashwitha Kassetty

Katherine Baumann

Ariella Amanuel

**HONOR CODE**

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence. I promise that I will submit only work that I personally create or that I contribute to group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of integrity and uphold the spirit of the Honor Code.

**SIGNATURES:** Ashwitha Venkata Kassetty, Ariella Amanuel, Katherine Baumann

**Table of Contents**

Task 1 pg.3-6

Task 2 pg. 7-13

Task 3 pg.14-23

**Task 1**

**Table USERS:**

CREATE TABLE USERS (

USRID int AUTO\_INCREMENT,

Name varchar(255) NOT NULL,

email varchar(255),

Phone varchar(255),

Type int NOT NULL,

PRIMARY KEY (USRID),

CONSTRAINT Type\_values CHECK ((Type=1 OR Type=2 OR Type=3))

);

I decided to use an incrementing, artificial key for the USRID field so the DBMS will provide a field for USRID. Also, I chose to not use the NOT NULL constraint for the email and Phone attributes in case a user does not have a phone number or email.

**Table AIRCRAFT:**

CREATE TABLE AIRCRAFT (

ARCID int AUTO\_INCREMENT,

Name varchar(255) NOT NULL,

ARLID varchar(2) NOT NULL,

PRIMARY KEY (ARCID)

);

I decided to use an incrementing, artificial key for the ARCID field so the DBMS will provide a unique field ARCID.

**Table AIRLINE:**

CREATE TABLE AIRLINE (

ARLID varchar(2),

Name varchar(255) NOT NULL,

PRIMARY KEY (ARLID)

);

**Table Airport:**

CREATE TABLE AIRPORT (

ARPID varchar(3),

Name varchar(255) NOT NULL,

PRIMARY KEY (ARPID)

);

**Table ARCCLASS:**

CREATE TABLE ARCCLASS (

ARCID int NOT NULL,

CLASS varchar(255) NOT NULL,

ChangeFee DECIMAL(8,2) NOT NULL,

NumofSeats int NOT NULL,

FOREIGN KEY (ARCID) REFERENCES AIRCRAFT (ARCID)

ON DELETE RESTRICT ON UPDATE CASCADE,

PRIMARY KEY (ARCID, CLASS)

);

For ChangeFee, I allowed up to 6 digits before the decimal point and 2 digits after the decimal point.

**Table ARLARP:**

CREATE TABLE ARLARP (

ARLID varchar(2),

ARPID varchar(3),

FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)

ON DELETE RESTRICT ON UPDATE CASCADE,

FOREIGN KEY (ARPID) REFERENCES AIRPORT (ARPID)

ON DELETE RESTRICT ON UPDATE CASCADE,

PRIMARY KEY(ARLID, ARPID)

);

**Table FLIGHT:**

CREATE TABLE FLIGHT (

ARLID varchar(2),

FLGID int AUTO\_INCREMENT,

Type int NOT NULL,

DepARPID varchar(3) NOT NULL,

DepTime time NOT NULL,

DesARPID varchar(3) NOT NULL,

DesTime time NOT NULL,

ARCID int NOT NULL,

FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)

ON DELETE RESTRICT ON UPDATE CASCADE,

PRIMARY KEY(ARLID,FLGID),

CONSTRAINT Type\_flvalues CHECK ((Type=1 OR Type=0))

);

I decided to use an incrementing, artificial key for the FLGID field so the DBMS will provide a unique value for FLGID.

**Table FLGCLASS:**

CREATE TABLE FLGCLASS (

ARLID varchar(2),

FLGID int,

CLASS varchar(255),

Fare decimal(8,2) NOT NULL,

FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)

ON DELETE RESTRICT ON UPDATE CASCADE,

FOREIGN KEY (FLGID) REFERENCES FLIGHT (FLGID)

ON DELETE RESTRICT ON UPDATE CASCADE,

PRIMARY KEY(ARLID, FLGID, CLASS)

);

For Fare, I allowed up to 6 digits before the decimal point and 2 digits after the decimal point.

**Table FLGDAYS:**

CREATE TABLE FLGDAYS (

ARLID varchar(2),

FLGID int,

DAY varchar(3),

FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)

ON DELETE RESTRICT ON UPDATE CASCADE,

FOREIGN KEY (FLGID) REFERENCES FLIGHT (FLGID)

ON DELETE RESTRICT ON UPDATE CASCADE,

CONSTRAINT DAY\_values CHECK ((DAY='Mon' OR DAY='Tue' OR DAY='Wed' OR DAY='Thu' OR DAY='Fri' OR DAY='Sat' OR DAY='Sun')),

PRIMARY KEY(ARLID, FLGID, DAY)

);

**Table TICKET:**

CREATE TABLE TICKET (

TCKID int AUTO\_INCREMENT,

USRID int NOT NULL,

Type int NOT NULL,

TotalFare decimal(8,2) NOT NULL,

PurchaseDateTime timestamp NOT NULL,

PRIMARY KEY (TCKID),

CONSTRAINT Type\_tckvalues CHECK ((Type=1 OR Type=2))

);

For TotalFare, I allowed up to 6 digits before the decimal point and 2 digits after the decimal point.

**Table SEQUENCE:**

CREATE TABLE SEQUENCE (

TCKID int,

ARLID varchar(2),

FLGID int,

CLASS varchar(255) NOT NULL,

TravelDate date NOT NULL,

FOREIGN KEY (TCKID) REFERENCES TICKET (TCKID)

ON DELETE RESTRICT ON UPDATE CASCADE,

FOREIGN KEY (ARLID) REFERENCES AIRLINE (ARLID)

ON DELETE RESTRICT ON UPDATE CASCADE,

FOREIGN KEY (FLGID) REFERENCES FLIGHT (FLGID)

ON DELETE RESTRICT ON UPDATE CASCADE,

PRIMARY KEY(TCKID,ARLID,FLGID)

);

**Task 2**

**USERS:**

Command:

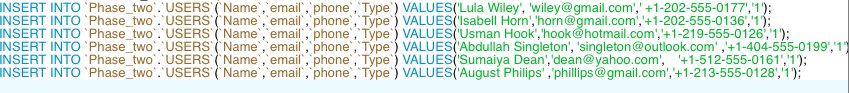
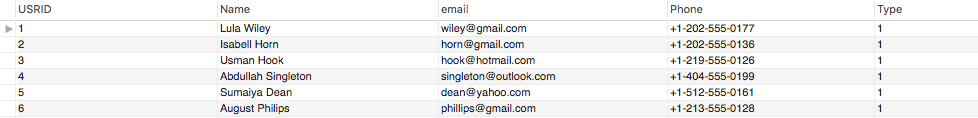
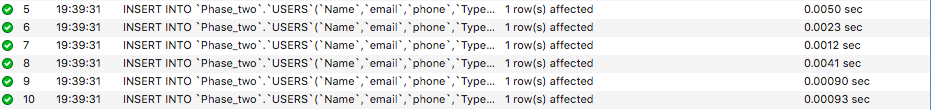


Table:



Action Output:



**AIRCRAFT:**

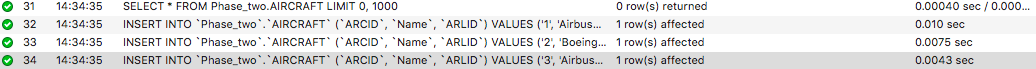
Command:



Table:



Action Output:



**AIRLINE:**

Command:

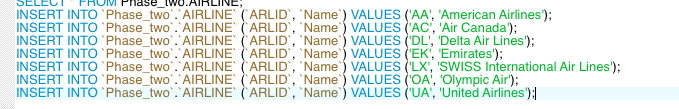


Table:



Action Output:

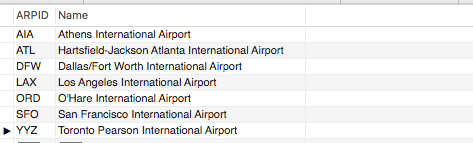


**AIRPORT:**

Command:



Table:



Action Output:



**ARCCLASS:**

Command:

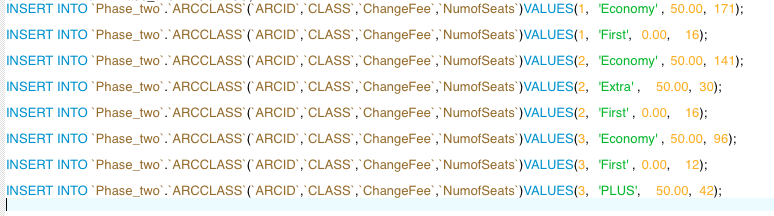
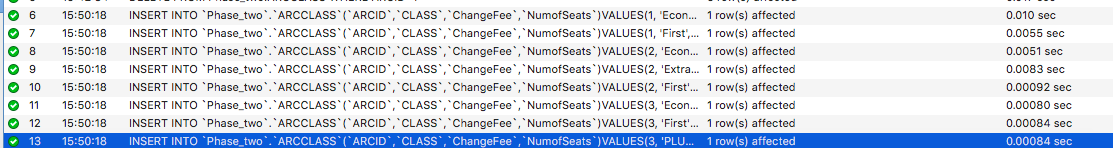


Table:



Action Output:

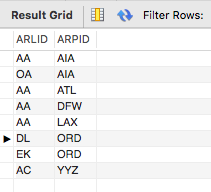


**ARLARP:**

Command:



Table:



Action Output:



**FLIGHT:**

Command:

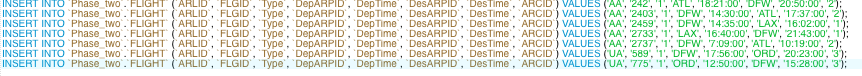
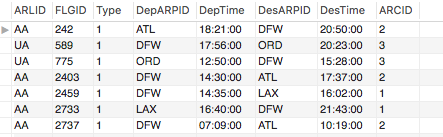
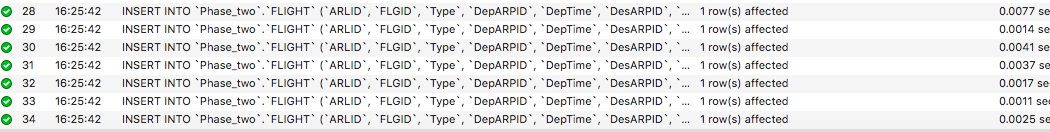


Table:



Action Output:



**FLGCLASS:**

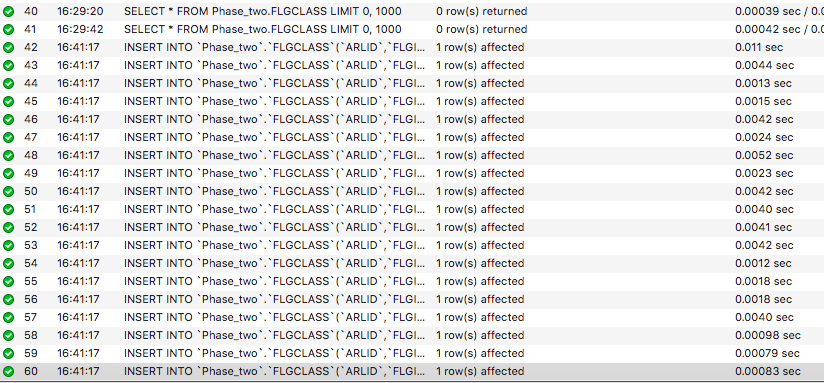
Command:



Table:



Action Output:

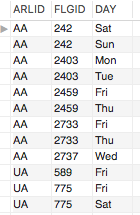


**FLGDAYS:**

Command:



Table:



Action Output:



**TICKET:**

Command:



Table:



Action Output:



**SEQUENCE:**

Command:

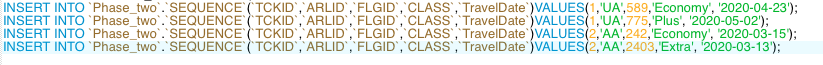
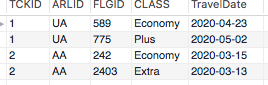
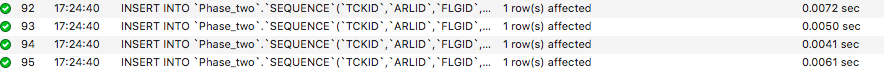


Table:



Action Output:



**Task 3**

**1) Insert yourself as a New Customer. Do not provide the customer id in your query.**

I used my UTA email and a fake phone number for the email and phone values.

**Query Text:**

INSERT INTO USERS(Name,email,Phone,Type)

VALUES('Katie Baumann','[katherine.baumann@mavs.uta.edu](mailto:katherine.baumann@mavs.uta.edu)','+1-123-456-7890',1);

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**2) Update your phone number to +1-837-721-8965**

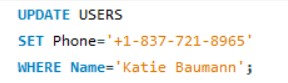
**Query Text:**

UPDATE USERS

SET Phone='+1-837-721-8965'

WHERE Name='Katie Baumann';

**Query Screenshot:**

****

**Query Output:**

N/A

**Action Output:**

****

**3) The value of TICKET.TotalFare for both records is wrong, and you need to write commands to update it.**

**a) Write an SQL query that returns for every line of the table SEQUENCE the flight fare.**

**Query Text:**

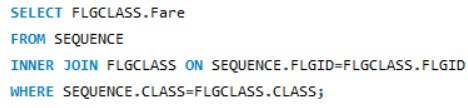
SELECT FLGCLASS.Fare

FROM SEQUENCE

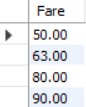
INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID

WHERE SEQUENCE.CLASS=FLGCLASS.CLASS;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**b) Then, write another query that will return the total fare per ‘SEQUENCE’ record.**

**Query Text:**

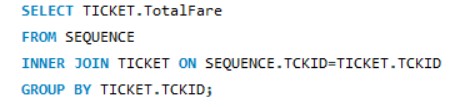
SELECT TICKET.TotalFare

FROM SEQUENCE

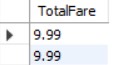
INNER JOIN TICKET ON SEQUENCE.TCKID=TICKET.TCKID

GROUP BY TICKET.TCKID;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**c) Finally, run an update command to correct the wrong values.**

**Query Text:**

UPDATE TICKET

INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID

INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID

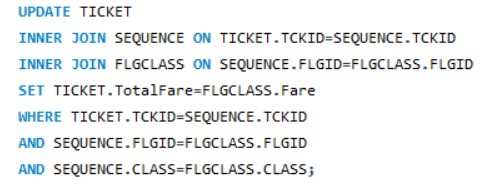
SET TICKET.TotalFare=FLGCLASS.Fare

WHERE TICKET.TCKID=SEQUENCE.TCKID

AND SEQUENCE.FLGID=FLGCLASS.FLGID

AND SEQUENCE.CLASS=FLGCLASS.CLASS;

**Query Screenshot:**

****

**Query Output:**

N/A

**Action Output:**



**d) Select all records from table ‘TICKET’.**

**Query Text:**

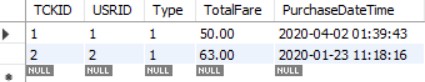
SELECT \*

FROM TICKET;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**



**4) Insert a new ‘TICKET’ record for you, as a customer, based on the information below:**

**Flight: AA2459,**

**Departure from: Dallas/Fort Worth International Airport,**

**Destination: Los Angeles International Airport,**

**Travel date: 05/29/2020**

**Class: Economy, One-way**

**Make sure that you will update with the correct information for both tables (‘TICKET’ and ‘SEQUENCE’). Show your work in steps.**

First, I inserted a record into the TICKET table using my USRID and the information provided from the question. I used the current\_timestamp() function to use the time I ran the command as the time I purchased the “ticket”. I also just used 9.99 for TotalFare as a placeholder until I updated said column with correct information.

**Query Text:**

INSERT INTO TICKET(TCKID,USRID,Type,TotalFare,PurchaseDateTime)

VALUES(3,8,1,9.99,current\_timestamp());

**Query Screenshot:**

****

**Query Output:**



**Action Output:**

****

Next, I inserted the rest of the information provided from the question into the SEQUENCE table.

**Query Text:**

INSERT INTO SEQUENCE(TCKID,ARLID,FLGID,CLASS,TravelDate)

VALUES(3,'AA',2459,'Economy','2020-05-29');

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

Finally, I updated the price in TICKET.TotalFare with the correct information.

**Query Text:**

UPDATE TICKET

INNER JOIN SEQUENCE ON TICKET.TCKID=SEQUENCE.TCKID

INNER JOIN FLGCLASS ON SEQUENCE.FLGID=FLGCLASS.FLGID

SET TICKET.TotalFare=FLGCLASS.Fare

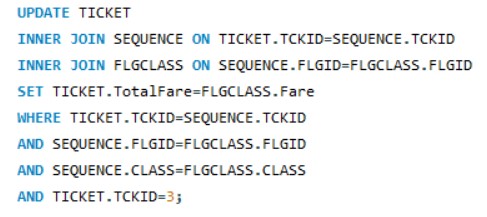
WHERE TICKET.TCKID=SEQUENCE.TCKID

AND SEQUENCE.FLGID=FLGCLASS.FLGID

AND SEQUENCE.CLASS=FLGCLASS.CLASS

AND TICKET.TCKID=3;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**5) Return for every flight, the airline id, the flight id, the departure and destination airport description, and the aircraft description. Furthermore, on the same query, return the type, but instead of 1, return ‘Domestic’. Order your results by the departure airport id.**

I created a nested JOIN on tables FLIGHT, AIRPORT (2x) and AIRCRAFT to achieve a combined table with all the required fields. I used CASE to ensure that “Domestic” is printed for type = 1.

**Query Text:**

SELECT F.ARLID, F.FLGID, ARP.Name AS “Departure Airport”, AP.Name AS “Destination Airport”, ARC.Name AS “Aircraft Name”,

CASE WHEN F.Type = 1 THEN “Domestic”

ELSE “International”

END AS “Type”

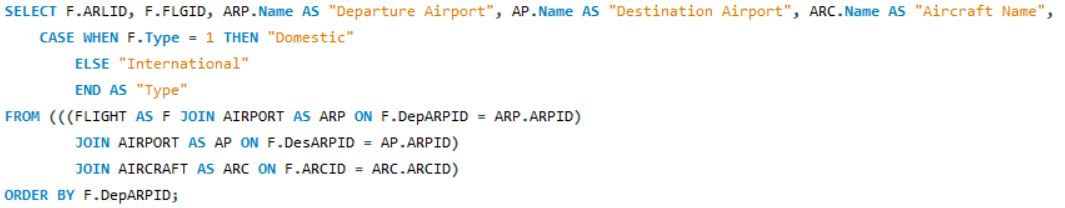
FROM (((FLIGHT AS F JOIN AIRPORT AS ARP ON F.DepARPID = ARP.ARPID)

JOIN AIRPORT AS AP ON F.DesARPID = AP.ARPID)

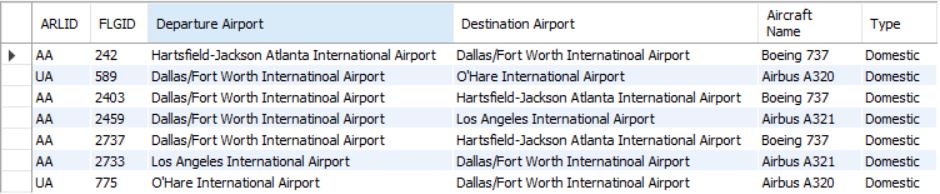
JOIN AIRCRAFT AS ARC ON F.ARCID = ARC.ARCID)

ORDER BY F.DepARPID;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**6) Present only for the economy class a prediction of an increased fare by 5%. In your results, you need to include all classes, not only the predicted economy fare. Note that this query will only present a prediction of 5% for the economy fare. It will not update any record in your database.**

I used the UNION command to combine the two cases mentioned in the question.

**Query Text:**

(SELECT CLASS, (Fare + (Fare\*0.05)) AS ‘Predicted Fare’

FROM FLGCLASS

WHERE CLASS = ‘Economy’)

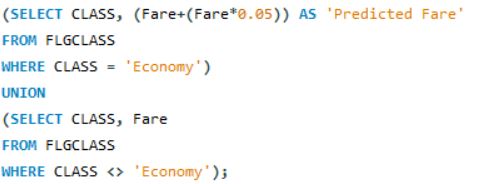
UNION

(SELECT CLASS, Fare

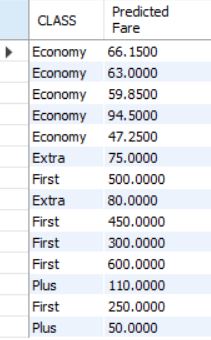
FROM FLGCLASS

WHERE CLASS <> ‘Economy’);

**Query Screenshot:**



**Query Output:**



**Action Output:**

****

**7) Return the airline ids and the airline description for those airlines that are not associated with any airport.**

I used nested queries with a NOT IN clause. I found the airline ids that were associated with an airport, and then used a NOT IN clause in the main query to get the expected output.

**Query Text:**

SELECT AL.ARLID, AL.Name

FROM AIRLINE AL, ARLARP AS AR

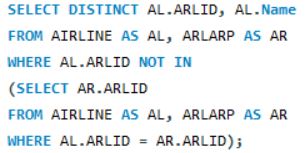
WHERE AL.ARLID NOT IN

(SELECT AR.ARLID

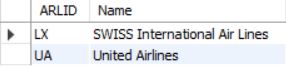
FROM ARLINE AS AL, ARLARP AS AR

WHERE AL.ARLID = AR.ARLID);

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**8) Return the airline id, airline description, and the associated airports’ ids and descriptions. For**

**the airlines that are not associated with any airport, instead of having blank cells, type ‘N/A’. (Hint: There is a DBMS function that replaces empty values).**

I created a nested LEFT OUTER JOIN on tables AIRPORT, ARLARP, and AIRPORT to produce a table that was combined by the airline ids and airport ids. I used the IFNULL() function to replace any empty values in airport id and airport name. I also renamed the column names of airline id, airline description, airport id, and airport description to Airline ID, Airline Name, Airport ID, Airport Name respectively.

**Query Text:**

SELECT AR.ARLID AS “Airport ID”, AR.Name AS “Airport Name”, IFNULL(AP.ARPID, “N/A”) AS “Airport ID”, IFNULL(AP.Name, “N/A”) AS “Airport Name”

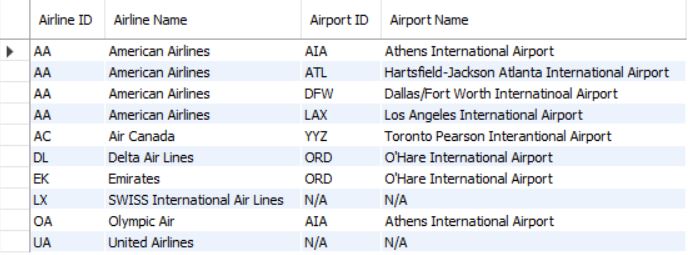
FROM ((AIRLINE AS AR LEFT OUTER JOIN ARPLARP AS ARP ON AP.ARLID = ARP.ARLID)

LEFT OUTER JOIN AIRPORT AS AP ON ARP.ARPID = AP.ARPID);

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**9) Write a query that will calculate for all tickets how many days left until the flight date. Ignore past tickets. (Hint: Do not hardcode today’s day. There is a DBMS functionality that automatically returns the today’s date).**

I used the DBMS function CURDATE() to obtain the current date.

**Query Text:**

SELECT S.TravelDate - CURDATE() AS ‘Days Left’

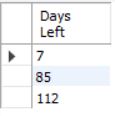
FROM SEQUENCE S, TICKET T

WHERE S.TCKID = T.TCKID AND (S.TravelDate-CURDATE()) >= 0;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****

**10) Write a query that will return the airline id, flight id, and all aircraft classes, along with the number of seats, the change fee and the fare. Order your results by the aircraft id.**

**Query Text:**

SELECT FG.ARLID, FG.FLGID, FG.CLASS,AC.NumofSeats, AC.ChangeFee, FG.Fare

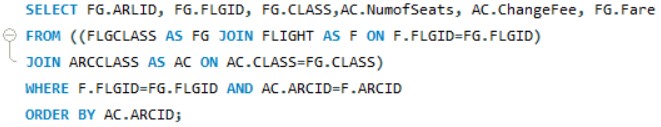
FROM ((FLGCLASS AS FG JOIN FLIGHT AS F ON F.FLGID=FG.FLGID)

JOIN ARCCLASS AS AC ON AC.CLASS=FG.CLASS)

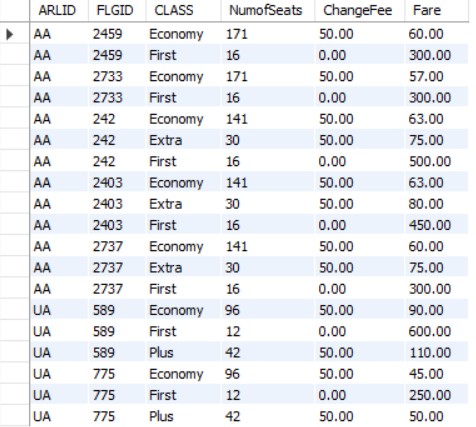
WHERE F.FLGID=FG.FLGID AND AC.ARCID=F.ARCID

ORDER BY AC.ARCID;

**Query Screenshot:**

****

**Query Output:**

****

**Action Output:**

****